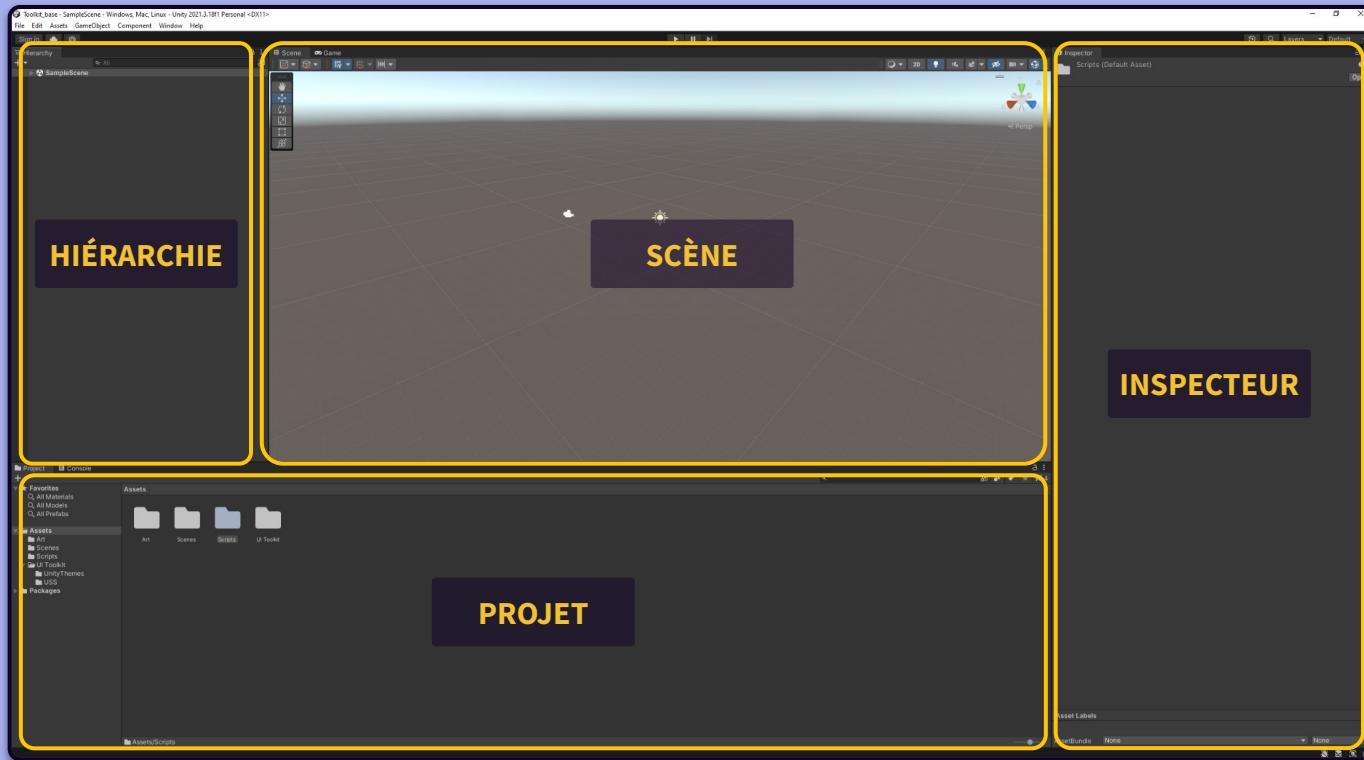


Créer sa première application 2D sur Unity



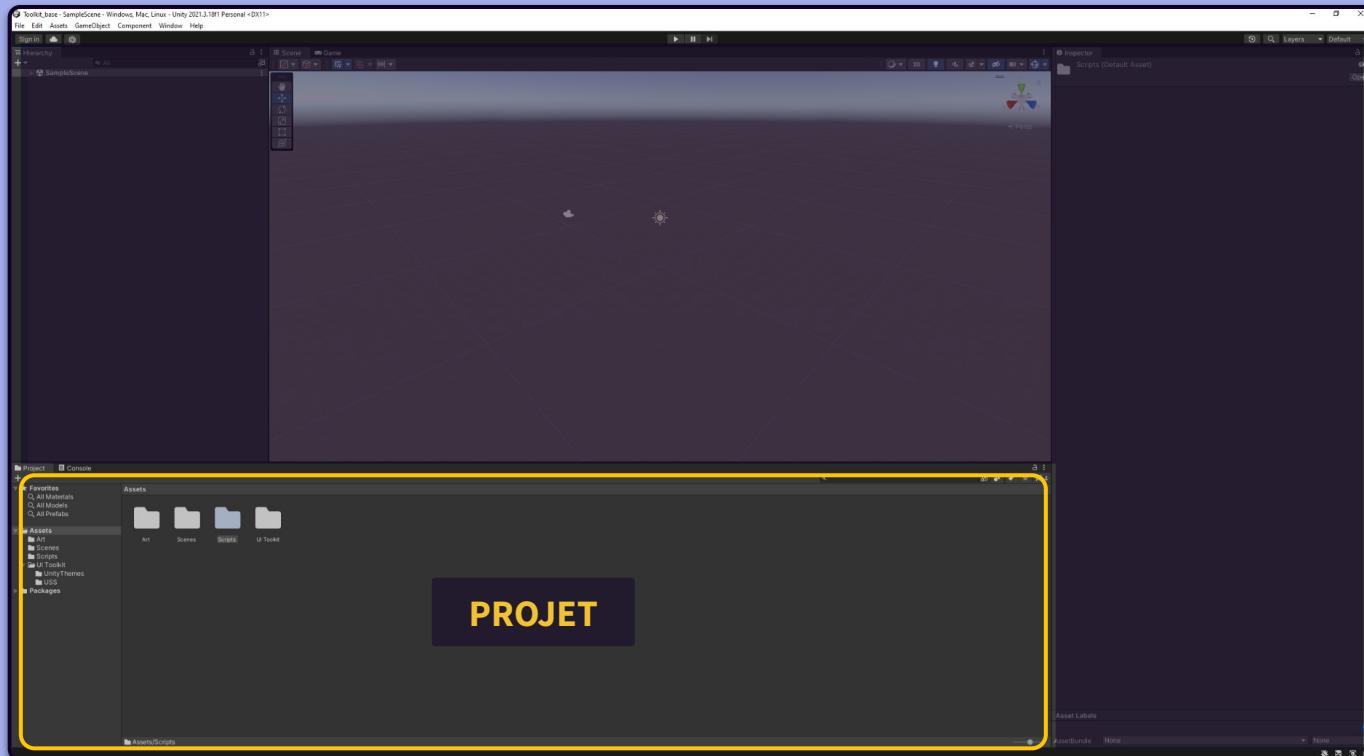
L'interface

se familiariser avec l'interface utilisateur de Unity (4 principales sections)



Section PROJET

contient tous les éléments qui apparaissent dans l'application



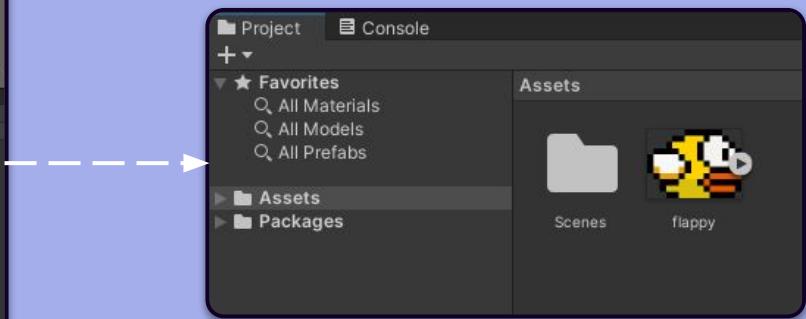
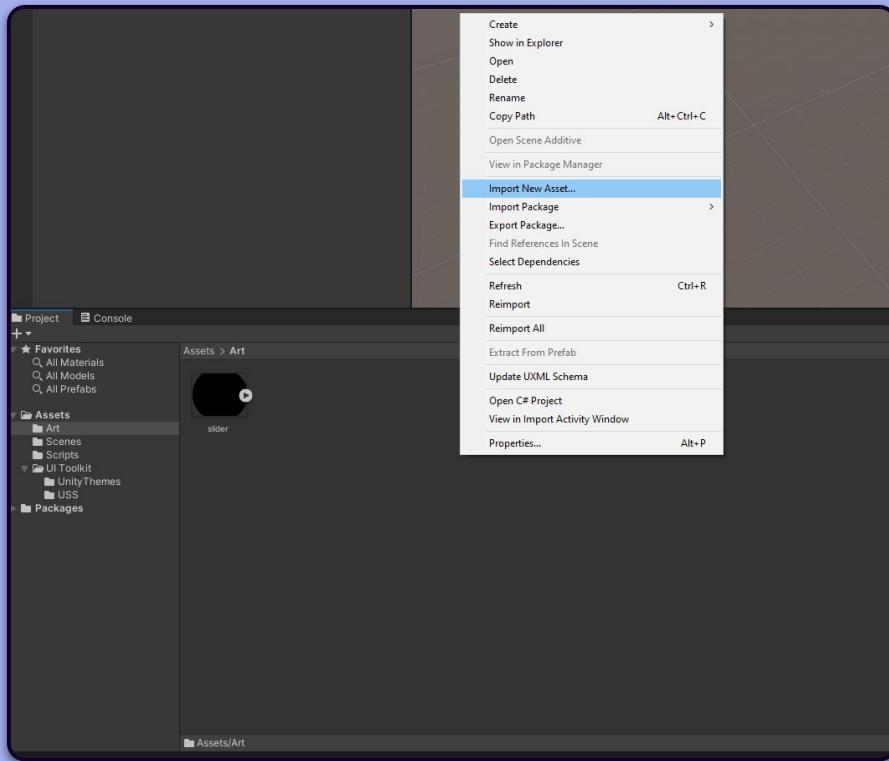
Section PROJET

contient tous les éléments qui apparaissent dans l'application



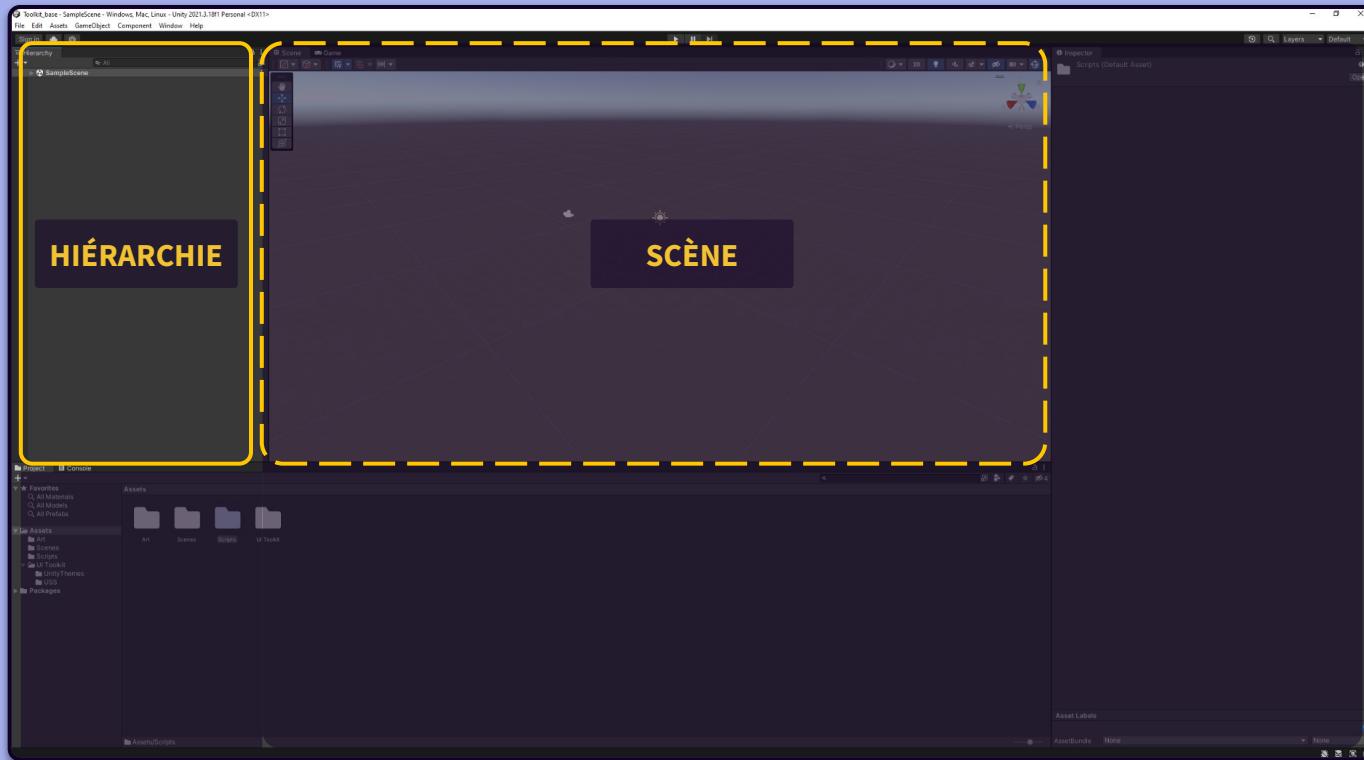
Ajouter un élément au projet

clic droit dans la section projet > Import New Asset...



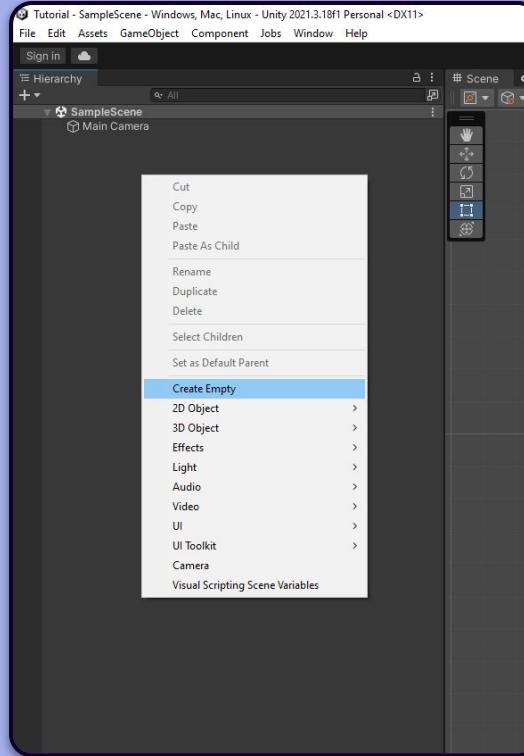
Section HIÉRARCHIE

contient tous les éléments qui apparaissent dans la **scène** actuelle



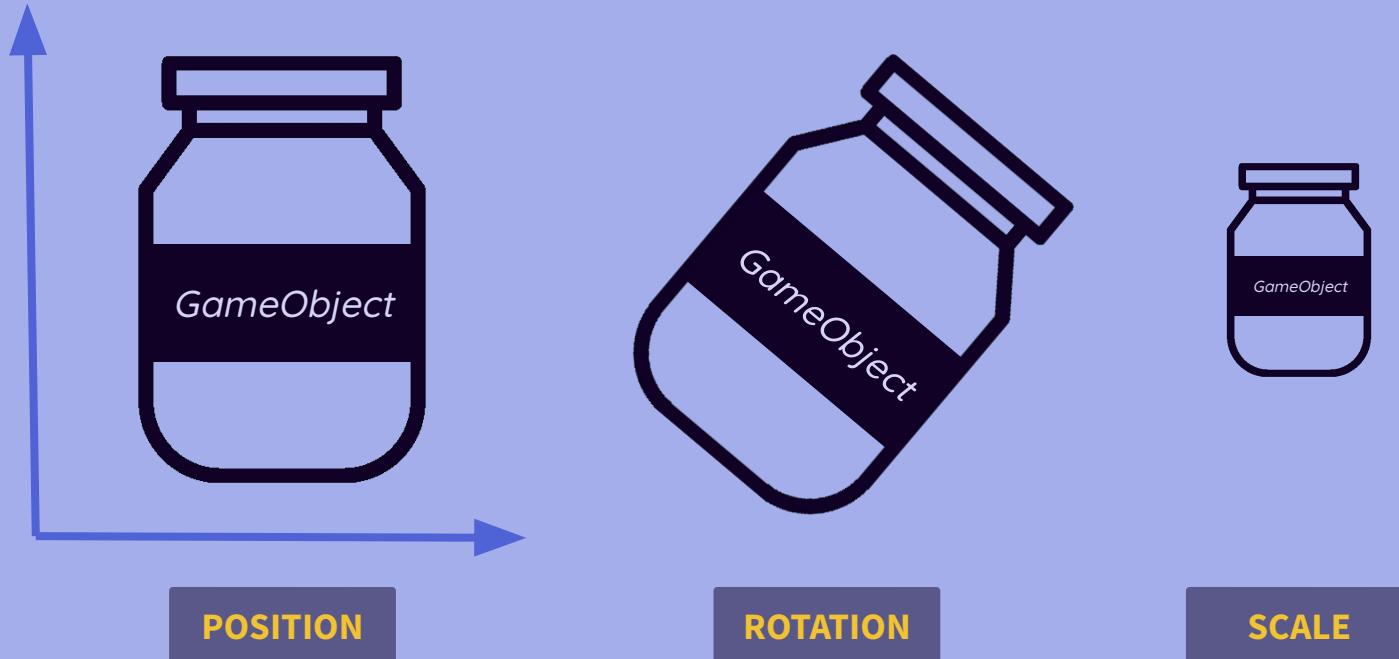
Créer un nouveau *GameObject*

clic droit dans la section HIÉRARCHIE > *Create Empty*



Qu'est ce qu'un *GameObject* ?

Un *GameObject* est un **conteneur** invisible avec une position dans l'espace, une rotation et une échelle via le composant *Transform*



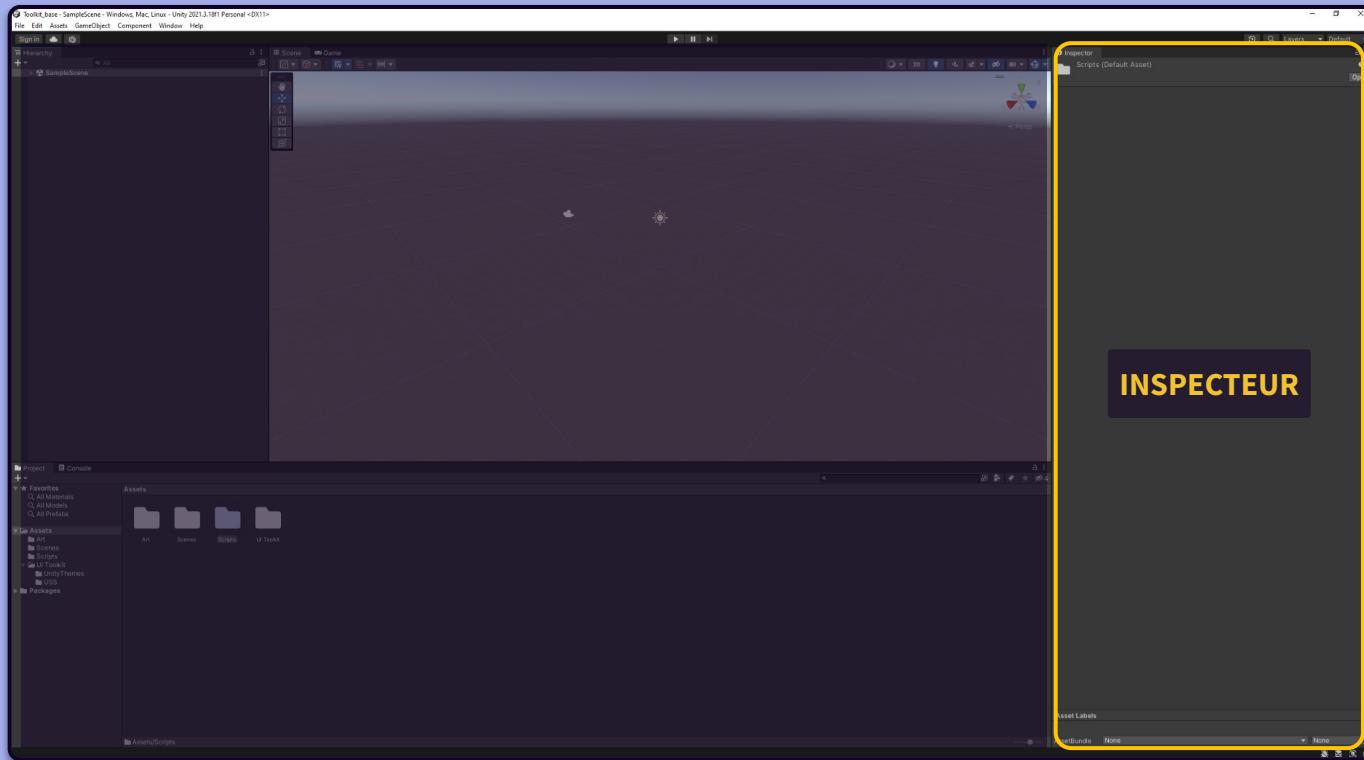
Qu'est ce qu'un *GameObject* ?

Un *GameObject* intègre des composants permettant de lui ajouter des propriétés supplémentaires



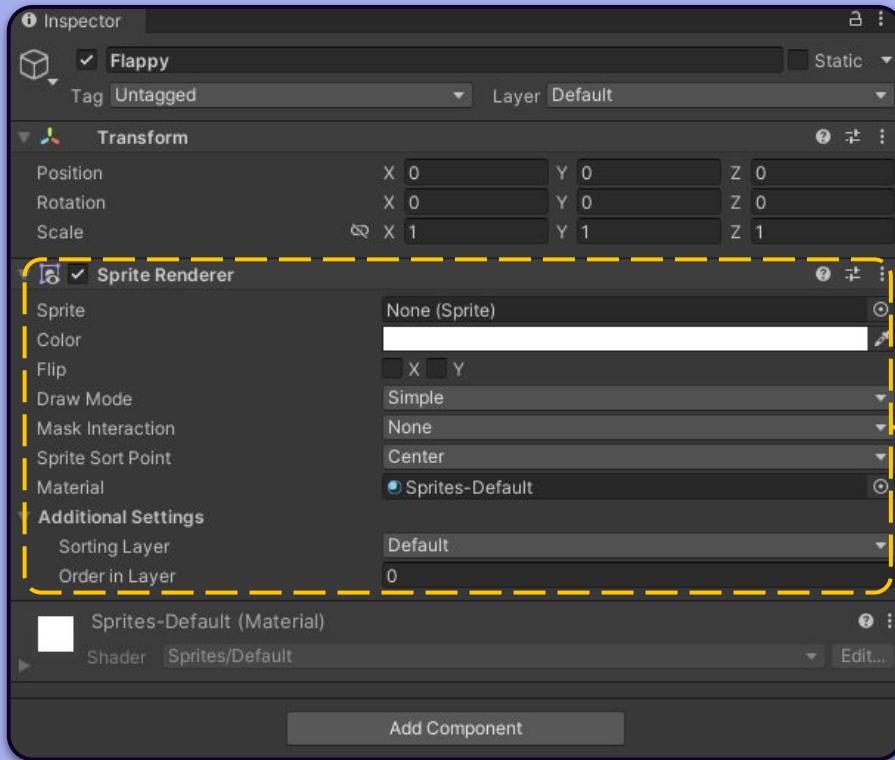
Section INSPECTEUR

affiche les composants des *GameObjects* et leurs propriétés



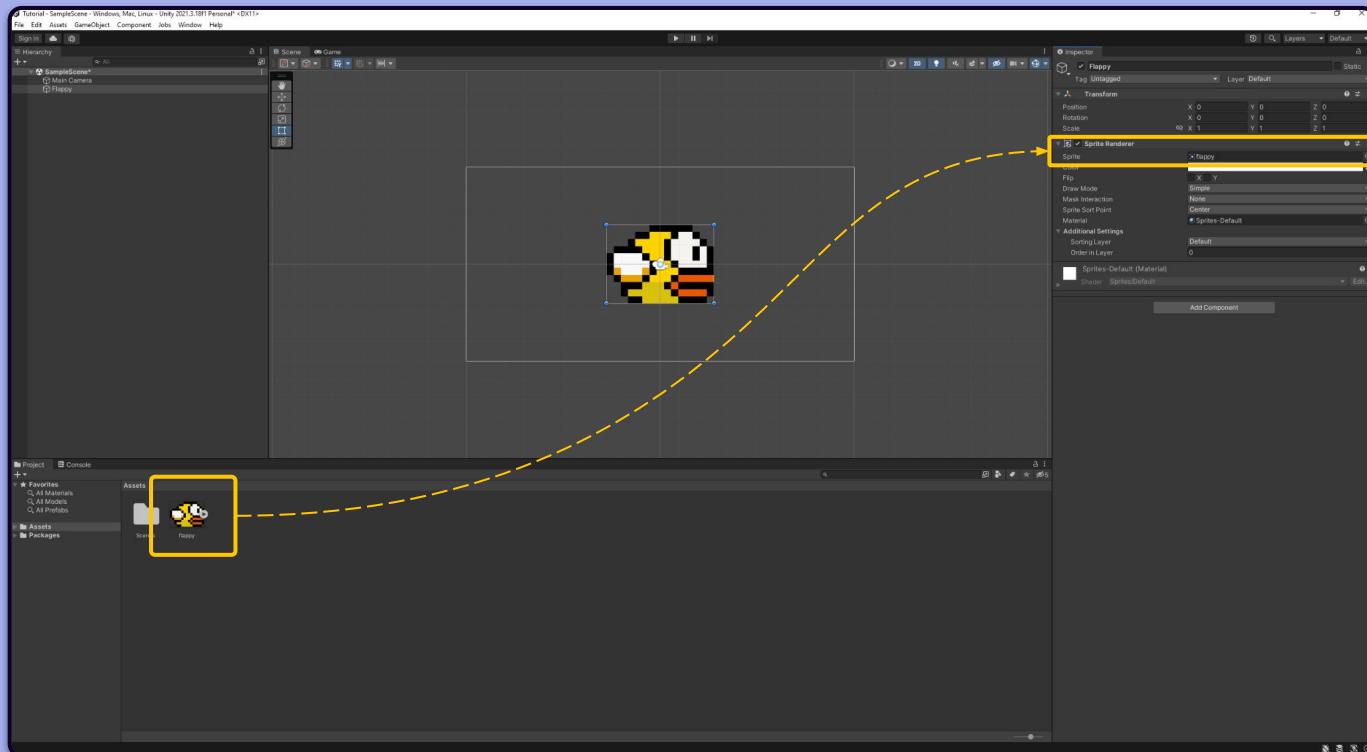
Ajouter un composant

cliquer sur *Add component > Rendering > Sprite Renderer*



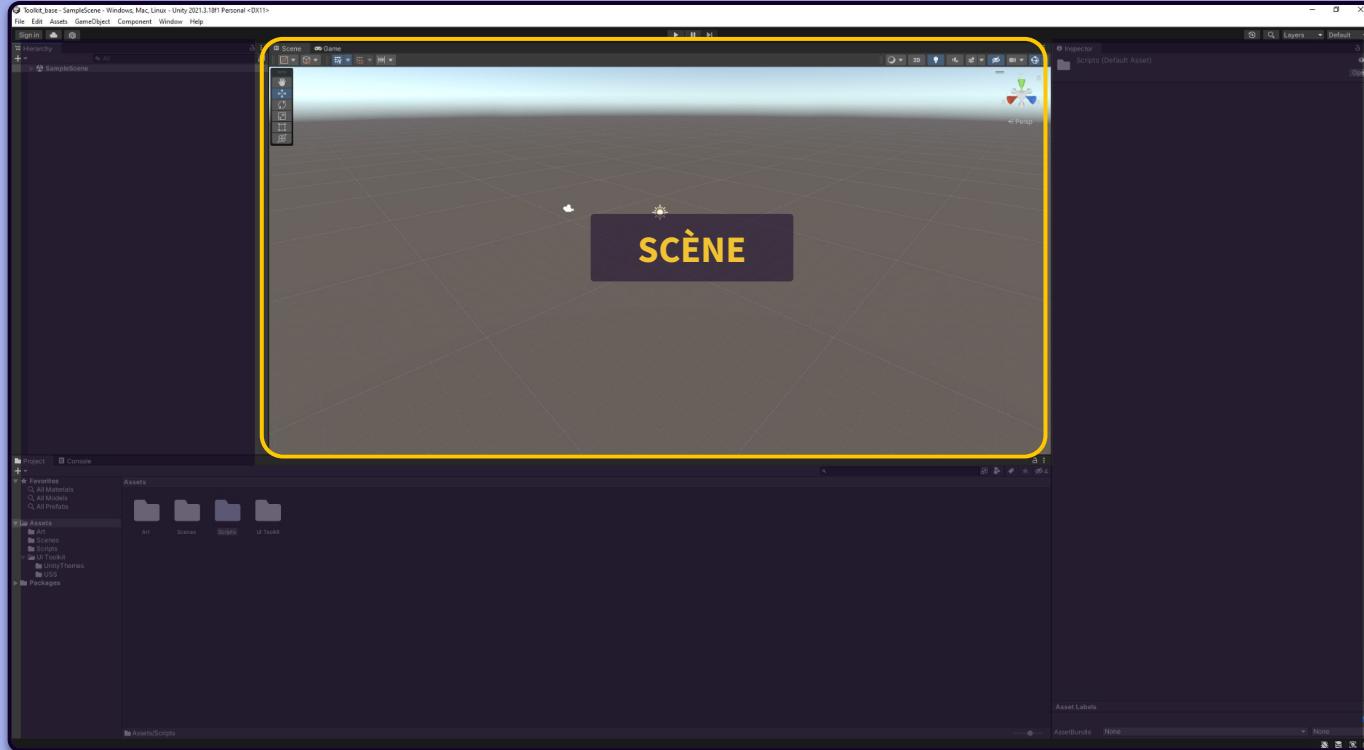
Modifier une propriété d'un composant

glisser-déposer l'image importée dans la section *Sprite* du composant



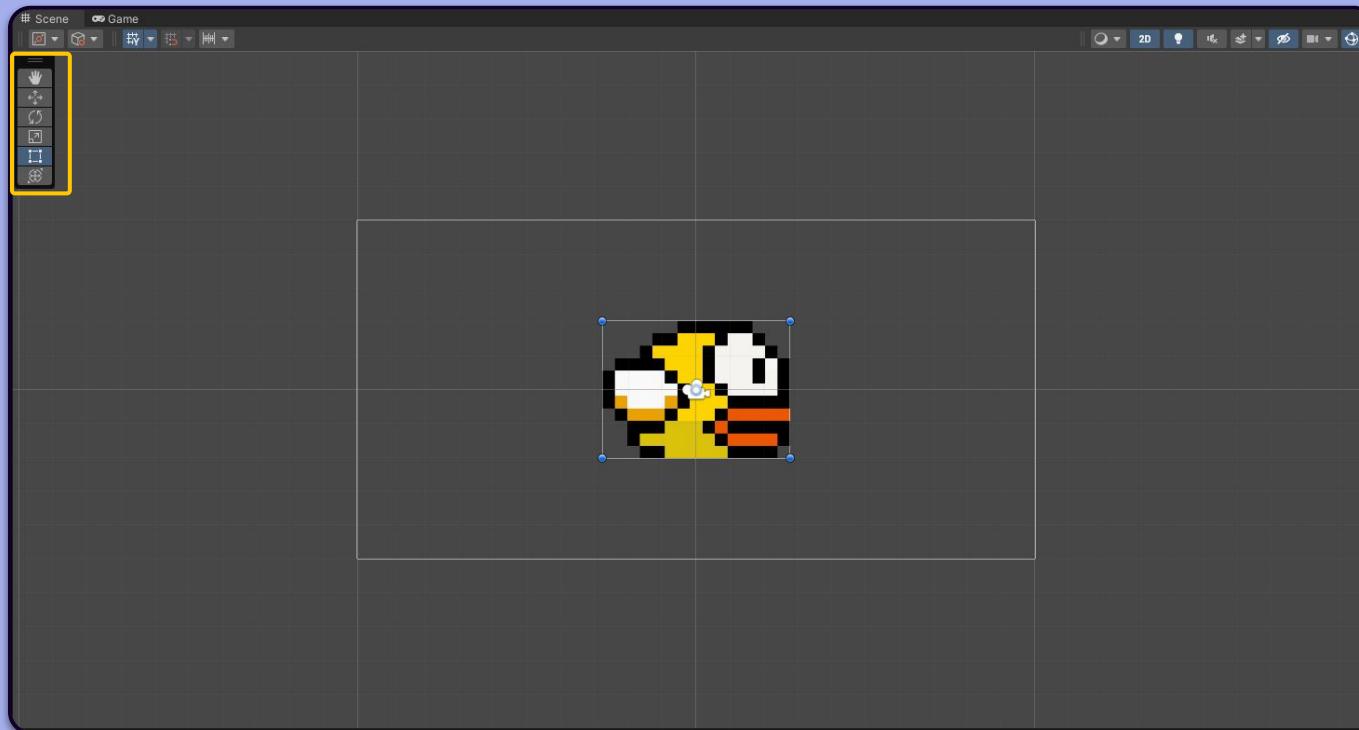
Section SCÈNE

affiche les représentations de nos *GameObjects* actifs



Section SCÈNE

offre des outils pour se déplacer, changer la position, la rotation et l'échelle de nos *GameObjects* présents dans la scène



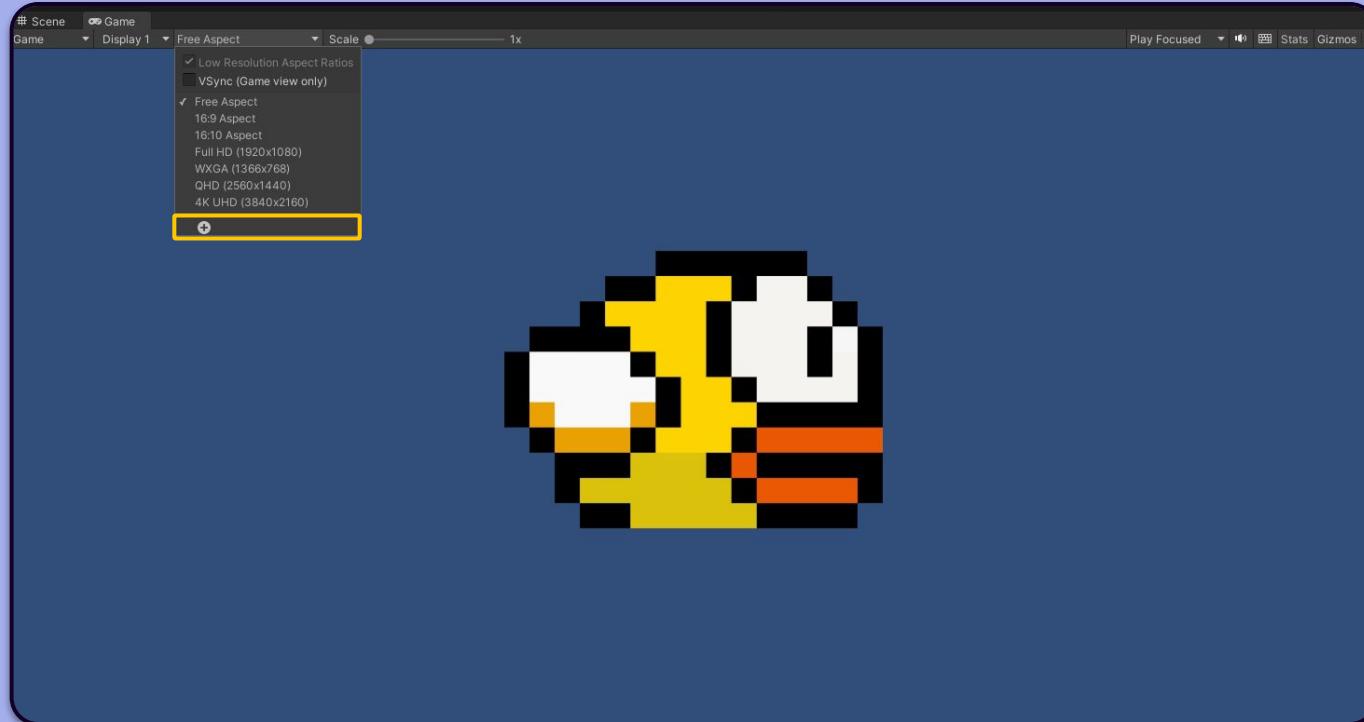
Section SCÈNE

l'onglet *Game* affiche le rendu de notre application via notre caméra principale de notre projet



Ajouter une résolution

cliquer sur *Free Aspect* > (+) > 750x1334 (Résolution de l'Iphone 7)



Ajouter l'arrière plan et le sol

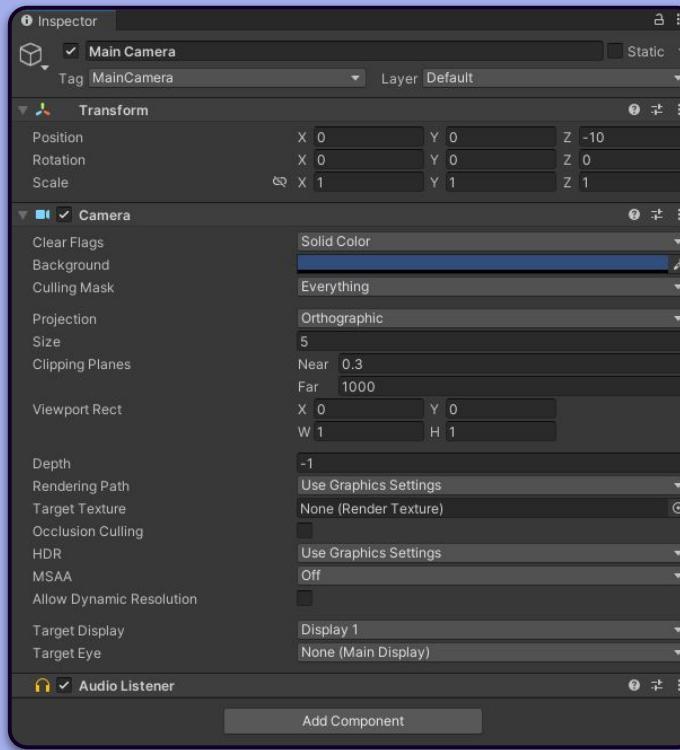
à vous de jouer, importer et ajouter à votre application les *GameObjects* adéquats



NB : La propriété *Order in Layer* du composant *Sprite Renderer* permet d'organiser la superposition des *sprites*

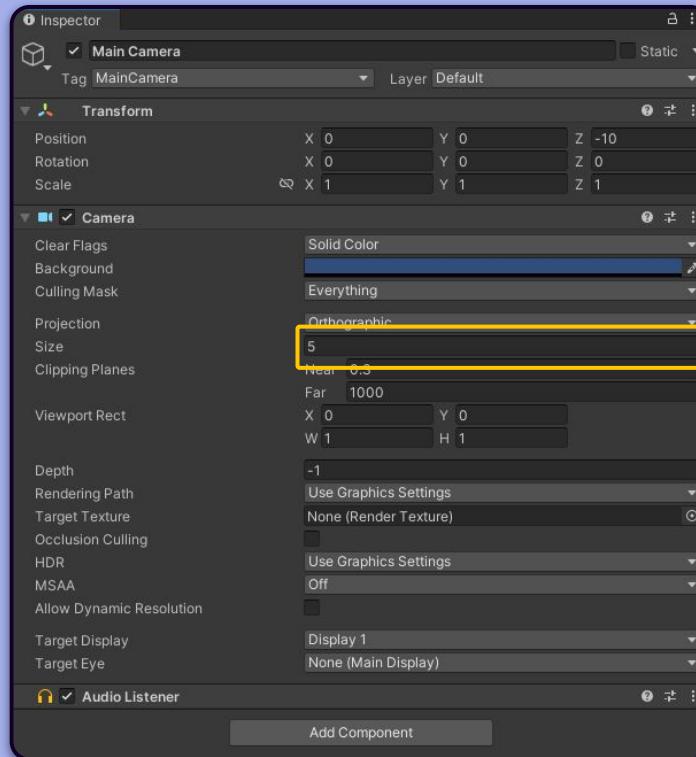
Caméra principale

le *GameObject* avec le Tag *MainCamera* et le composant *Camera* représente la caméra principale de notre application par laquelle l'utilisateur voit l'application



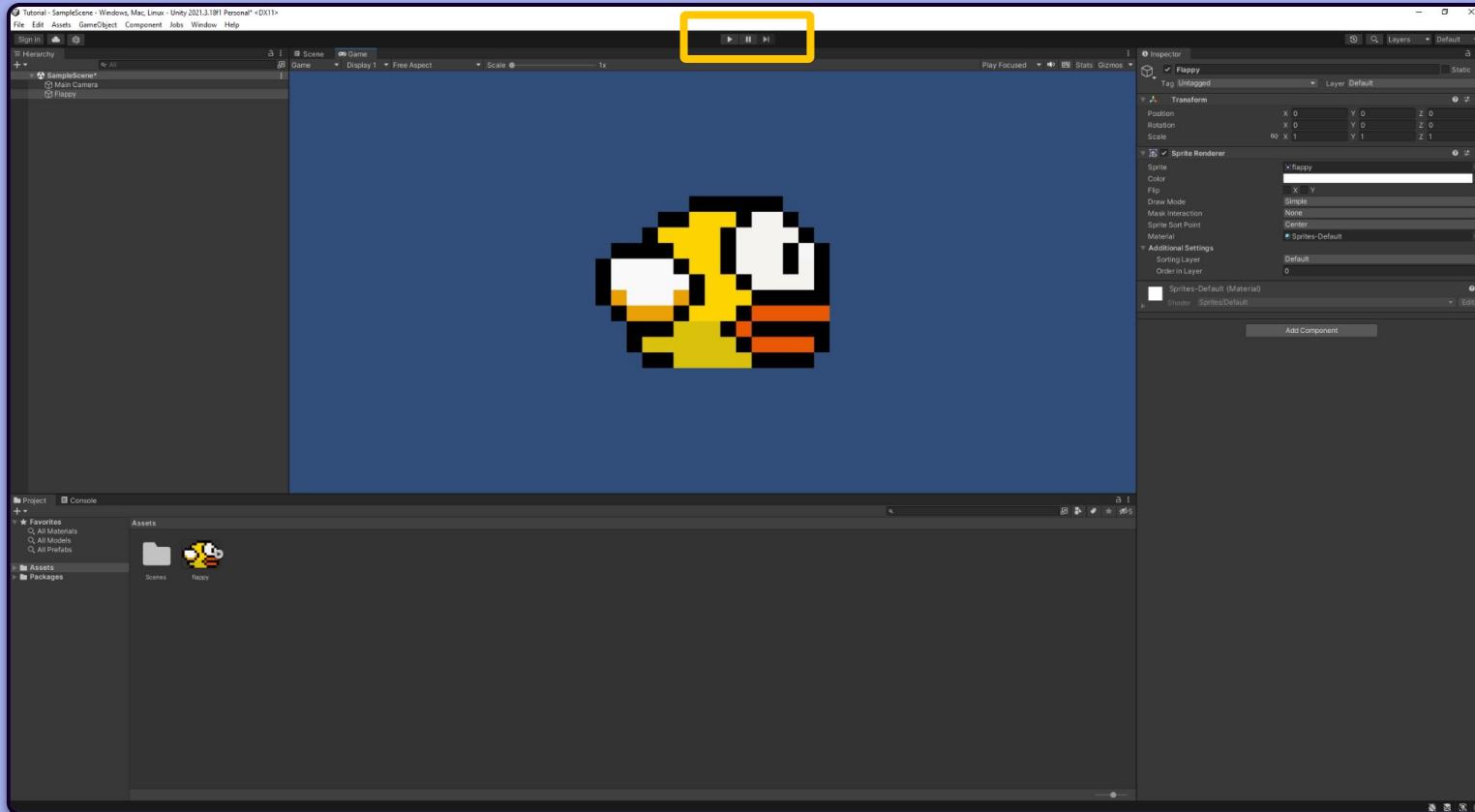
Modifier la propriété *size*

ajuster la propriété *Size* (zoom/dézoom) pour l'adapter à votre arrière plan (~2.5)



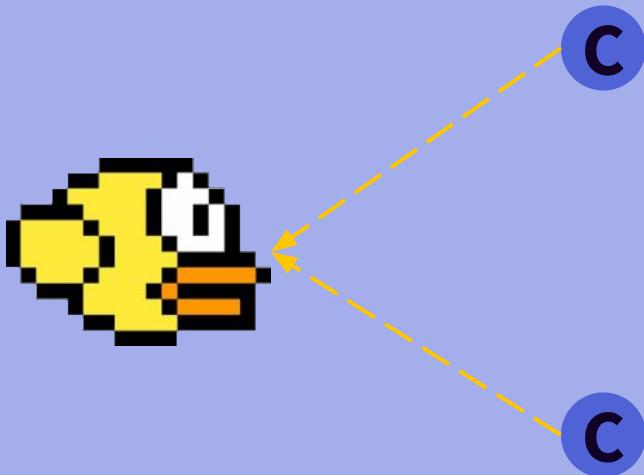
Lancer votre application

cliquer sur le bouton *Play* pour lancer le rendu de votre application



La physique et la programmation

ajouter un effet de gravité à *Flappy* et un nouveau composant personnalisé



PHYSICS

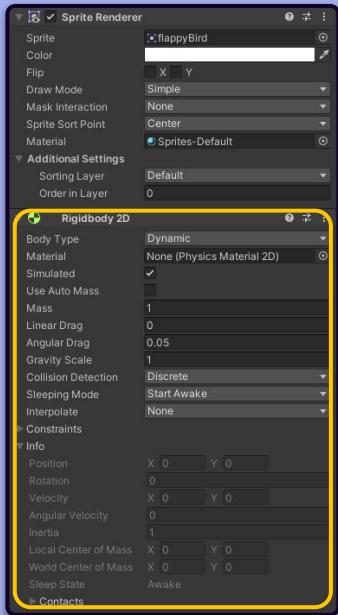
Ajout d'un composant permettant d'appliquer une gravité à notre GameObject

SCRIPT

Ajout d'un composant personnalisé (script) permettant à notre Flappy de voler en appuyant sur la barre espace

Ajouter le composant *RigidBody2D*

cliquer sur *Add component > Physics 2D > RigidBody 2D*



Notre *GameObject* est désormais un objet physique affecté par la gravité

Lancer votre application et observer

La propriété ***Linear Drag*** permet d'ajouter un effet de friction à notre *GameObject* pour ralentir sa chute (~1 si la chute est trop rapide)

La gravité applique tous les 50Hz par défaut (50 fois par seconde) la force suivante `Vector3(0, -9.81, 0)` à tous les *GameObjects* physique de la scène (*Edit > Project Settings > Physics > Gravity*)

Ajouter un composant personnalisé

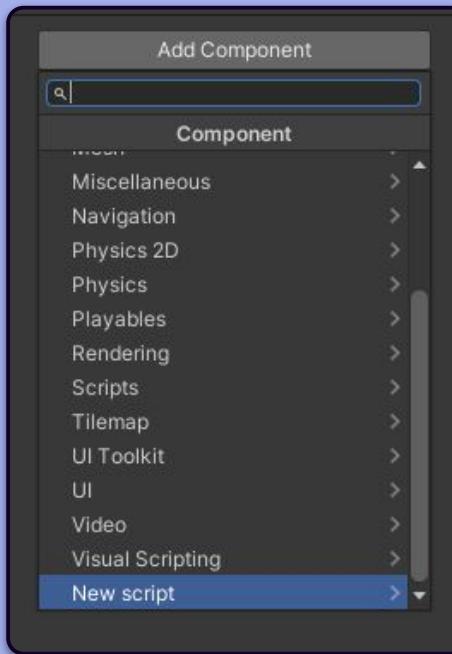
en utilisant la programmation C#



Donner une impulsion verticale à Flappy lorsque l'on appuie sur la barre espace

Créer un nouveau script pour Flappy

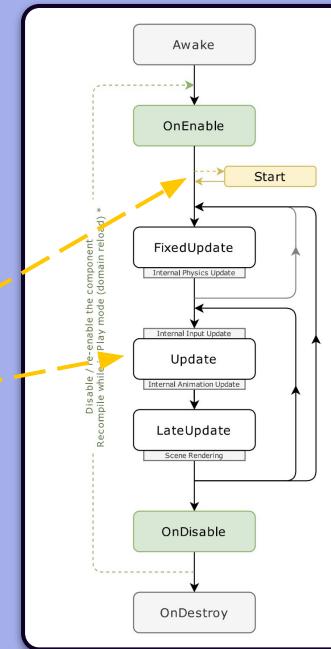
cliquer sur *Add component > New script*, nommer le *BirdScript* et double cliquer dessus



Anatomie d'un script

le cycle de vie d'un composant personnalisé

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BirdScript : MonoBehaviour
6  {
7
8      // Start is called before the first frame update
9      void Start()
10     {
11
12     }
13
14     // Update is called once per frame
15     void Update()
16     {
17
18     }
19
20 }
```

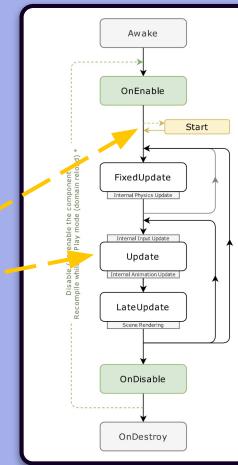


Start et *Update* font partie d'un ensemble de méthodes qui constitue le cycle de vie d'un composant Unity

Anatomie d'un script

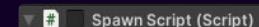
le cycle de vie d'un composant personnalisé

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BirdScript : MonoBehaviour
6  {
7
8      // Start is called before the first frame update
9      void Start()
10     {
11
12     }
13
14     // Update is called once per frame
15     void Update()
16     {
17
18     }
19
20 }
```



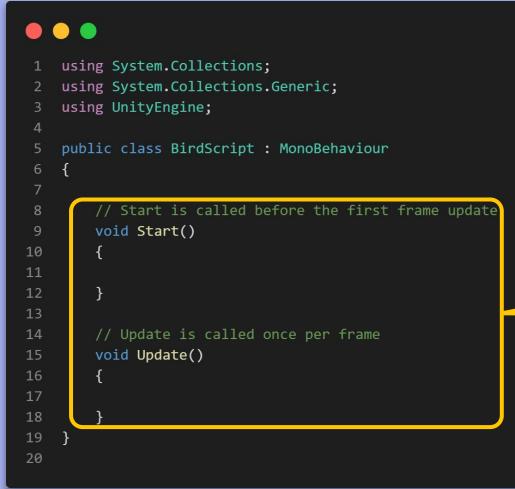
Le cycle de vie des méthodes respecte un ordre, la méthode *Awake* est appelée avant la méthode *OnEnable* elle même appelée avant la méthode *Start*

Tout comme la méthode *Awake*, **Start** est appelé exactement qu'une seule fois dans le cycle de vie d'un composant. Cependant, *Awake* est appelé quand le *GameObject* est initialisé indépendamment de l'activation du composant.

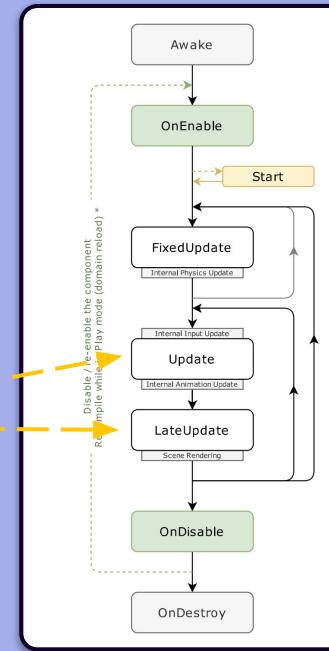


Anatomie d'un script

le cycle de vie d'un composant personnalisé



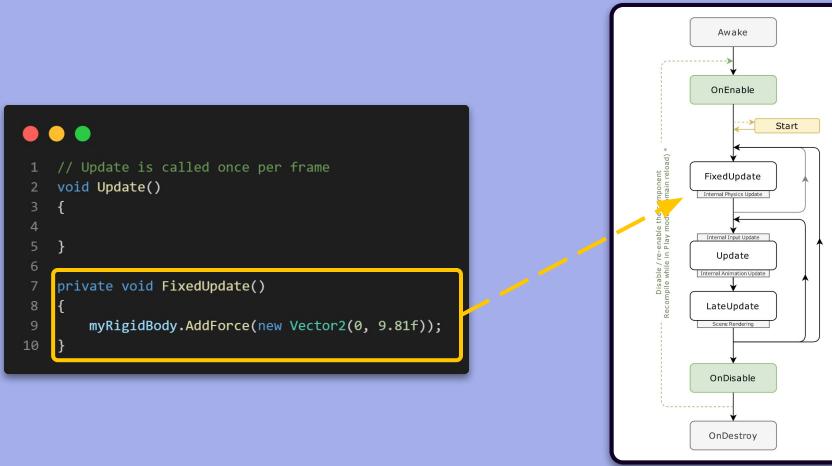
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BirdScript : MonoBehaviour
6  {
7
8      // Start is called before the first frame update
9      void Start()
10     {
11
12     }
13
14     // Update is called once per frame
15     void Update()
16     {
17
18     }
19
20 }
```



Les méthodes *Update* et *LateUpdate* sont appelées à chaque *frame* (tant que le script est actif), elles dépendent du taux de rafraîchissement (FPS) de l'utilisateur

Anatomie d'un script

le cycle de vie d'un composant personnalisé



La méthode **FixedUpdate** est indépendant du taux de rafraîchissement (FPS) de l'utilisateur, elle est appelée à la même fréquence que le système physique de Unity, soit 50 fois par seconde par défaut (20ms)

Elle est généralement employée pour gérer la physique d'un *GameObject*, dans cet exemple nous appliquons une force verticale vers le haut pour contrer l'effet de gravité

**AddForce()* modifie la vitesse par la valeur de *force * DeltaTime / masse*, cet effet dépend du taux de rafraîchissement du système physique et de la masse du *GameObject*

Première instruction C#

pour modifier le nom de notre *GameObject*

A partir du mot-clé *gameObject*, nous pouvons modifier les propriétés et les composants attachés à ce *GameObject*



Lancer l'application et observer le nom de votre *GameObject*

Le mot clé *gameObject* fait référence au *GameObject* qui possède ce script



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BirdScript : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start()
9      {
10         gameObject.name = "Bob the bird !";
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
```

Ajouter une référence à notre *GameObject*

par défaut, notre script ne peut pas discuter avec les autres composants du *GameObject*

Nous voulons modifier la propriété *velocity* du composant *Rigidbody2D* afin de donner une impulsion verticale à notre *GameObject*, par conséquent nous devons créer une référence à ce composant

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BirdScript : MonoBehaviour
6  {
7      public Rigidbody2D myRigidbody; // Line 7 highlighted with a yellow box
8
9      // Start is called before the first frame update
10     void Start()
11     {
12         gameObject.name = "Bob the bird !";
13     }
14
15     // Update is called once per frame
16     void Update()
17     {
18
19     }
20 }
```



Glisser-déposer le composant *Rigidbody2D* à l'emplacement *My Rigidbody*

Ajouter une référence à notre *GameObject*

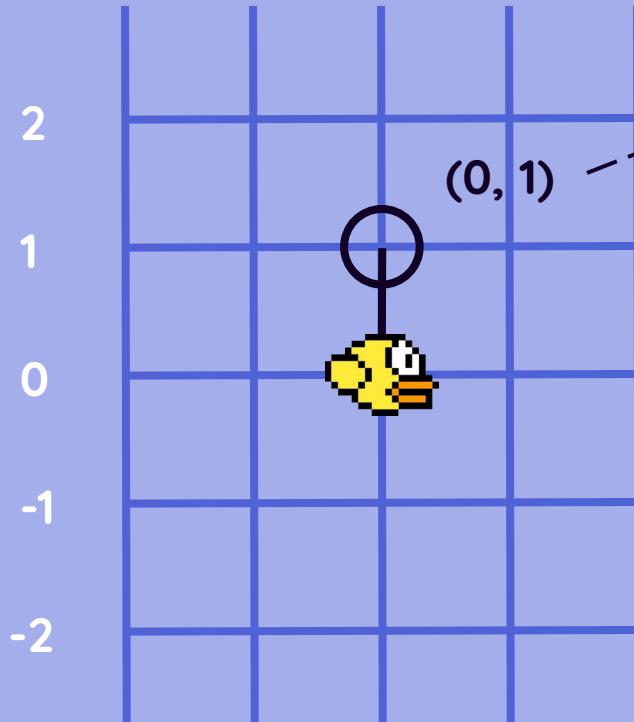
une alternative au glisser-déposer du composant (ici *Rigidbody2D*) est la méthode *GetComponent<T>()*

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class BirdScript : MonoBehaviour
6  {
7      public Rigidbody2D myRigidBody;
8
9      // Start is called before the first frame update
10     void Start()
11     {
12         gameObject.name = "Bob the bird !";
13         myRigidBody = GetComponent<Rigidbody2D>();
14     }
15
16     // Update is called once per frame
17     void Update()
18     {
19
20     }
21 }
22
```

Récupération du composant
Rigidbody2D du *gameObject* actuel

Modifier la *Velocity*

indiquer la direction de l'impulsion donnée à notre *GameObject*



Un vecteur de deux nombres, coordonnées *x* et *y*

● ● ●
1 // Update is called once per frame
2 void Update()
3 {
4 myRigidBody.velocity = Vector2.up * 5f;
5 }

Direction

Vitesse

$\text{Vector.up} == (0, 1)$

Lancer votre application et observer

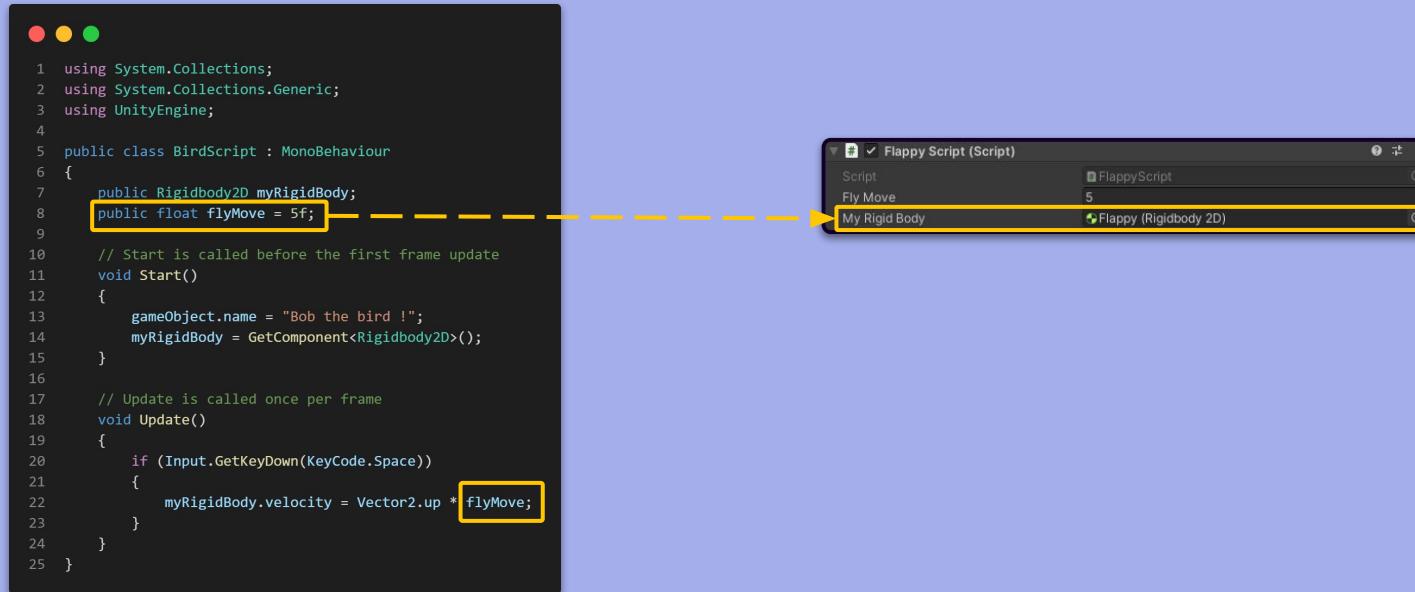
Réagir aux *inputs* de l'utilisateur

utilisation de l'*Input system* de Unity

```
● ● ●  
1 // Update is called once per frame  
2 void Update()  
3 {  
4     if (Input.GetKeyDown(KeyCode.Space))  
5     {  
6         myRigidBody.velocity = Vector2.up * 5f;  
7     }  
8 }
```

Lorsque l'on appuie sur la barre espace, la vitesse de notre *GameObject* est modifiée lui donnant ainsi une impulsion vers le haut

Ajouter une nouvelle propriété à notre composant pour faciliter sa paramétrisation en dehors du script



The diagram illustrates the process of adding a new property to a Unity script. On the left, a code editor window displays the `BirdScript.cs` file. A yellow dashed arrow points from the line `public float flyMove = 5f;` in the code to the `Fly Move` field in the Unity Inspector window on the right. Another yellow dashed arrow points from the `flyMove` variable in the `Update()` method to the `My Rigid Body` component in the Inspector, which is highlighted with a yellow border.

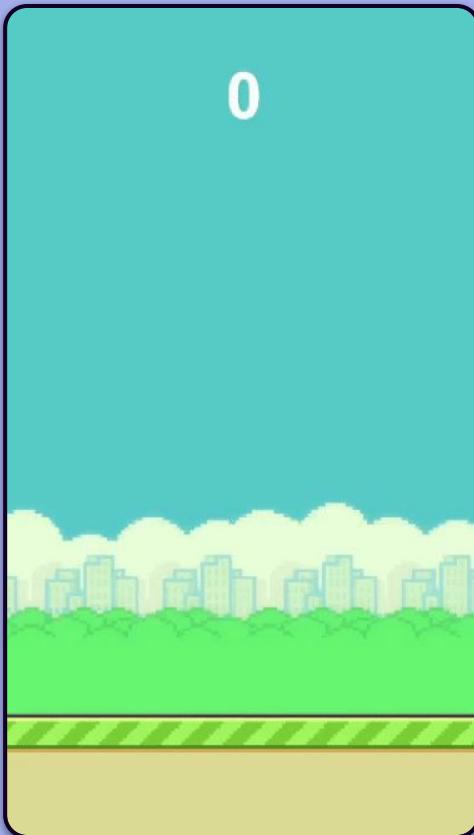
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class BirdScript : MonoBehaviour
6 {
7     public Rigidbody2D myRigidbody;
8     public float flyMove = 5f; // Line highlighted by a yellow box
9
10    // Start is called before the first frame update
11    void Start()
12    {
13        gameObject.name = "Bob the bird !";
14        myRigidbody = GetComponent<Rigidbody2D>();
15    }
16
17    // Update is called once per frame
18    void Update()
19    {
20        if (Input.GetKeyDown(KeyCode.Space))
21        {
22            myRigidbody.velocity = Vector2.up * flyMove; // Line highlighted by a yellow box
23        }
24    }
25 }
```

Flappy Script (Script)

Script	FlappyScript
Fly Move	5
My Rigid Body	Flappy (Rigidbody 2D)

Instancier des *GameObjects* et gestion l'UI

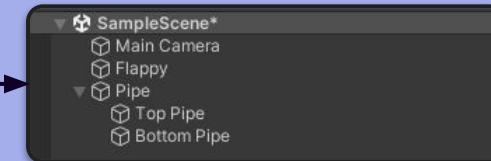
via la création de *prefab* et l'utilisation d'un canvas



Défilement horizontal des tuyaux

en imbriquant des *GameObjects*

Créer un *GameObject* nommé *Pipe* puis deux *GameObjects* imbriqués: *Top Pipe* et *Bottom Pipe*



Importer l'image du tuyau au projet et ajouter le composant *Sprite Renderer* aux *GameObjects* *Top* et *Bottom Pipe* afin d'inclure l'image du tuyau

Cocher la valeur Y de la propriété *Flip* pour le *Top Pipe* afin de l'inverser

Tester de modifier la position du *GameObject Pipe*, les *GameObjects* imbriqués se déplacent également

Défilement horizontal des tuyaux

ajouter un script au *GameObject Pipe*

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PipesMoveScript : MonoBehaviour
6  {
7      public float moveSpeed = 1f;
8
9      // Start is called before the first frame update
10     void Start()
11     {
12     }
13
14
15     // Update is called once per frame
16     void Update()
17     {
18         transform.position = transform.position + (Vector3.left * moveSpeed * Time.deltaTime);
19     }
20 }
```

(0, -1, 0)

Ajouter une variable pour représenter la vitesse de défilement

Réaliser un défilement horizontal vers la gauche

La méthode *Update()* est appelée à chaque *frame* et la valeur *Time.deltaTime* retourne le temps en seconde qui s'est écoulé depuis la dernière *frame*

La valeur de *Time.deltaTime* est plus élevée pour les utilisateurs avec un plus petit FPS, elle permet donc de normaliser la vitesse de défilement indépendamment du FPS des utilisateurs

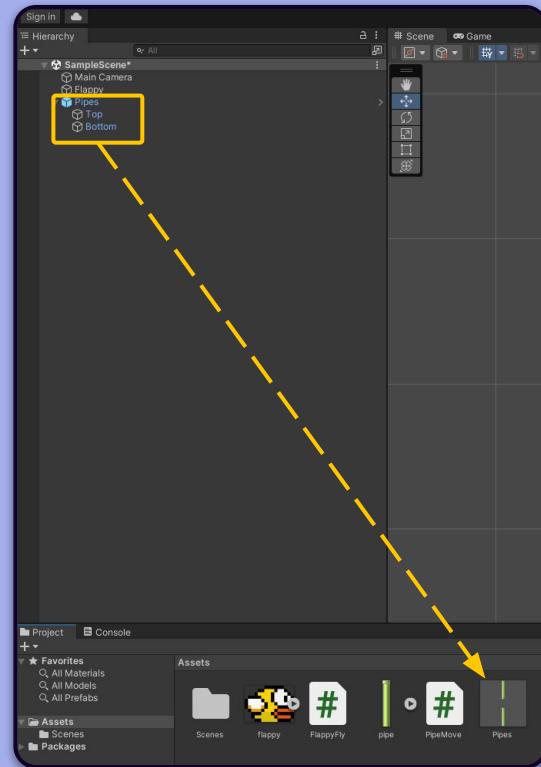
Créer un *prefab* de Pipe

permettra d'instancier ce *GameObject*

Une *prefab* est un *blueprint*, un moule d'un *GameObject* ce qui nous permet de créer des instances de ce *GameObject* à volonté tout en conservant ses enfants, ses composants et ses propriétés



Faites un glissé-déposé du *GameObject Pipes* vers votre dossier *Assets*



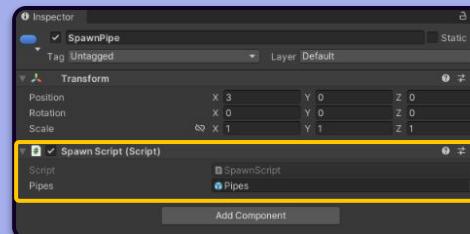
Un *GameObject* pour gérer l'instanciation de nos *Pipes*

créer un *GameObject* et lui associer un comportement personnalisé

Supprimer le *GameObject Pipe* de la hiérarchie et créer un nouveau *GameObject SpawnPipe* que vous pouvez positionner à droite de votre scène



Ajouter un script *SpawnScript* pour créer de nouveaux *Pipes* toutes les X secondes



Un *GameObject* pour gérer l'instanciation de nos *Pipes*

créer une fonction *NewPipe()* pour instancier de nouveaux *Pipes* à volonté

Mettre un *offset vertical* aléatoire pour modifier la hauteur des *Pipes* générés

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class SpawnScript : MonoBehaviour
6 {
7     public GameObject pipes;
8
9     float timeBetween = 2f;
10    float currentTime = 0f;
11
12    // Start is called before the first frame update
13    void Start()
14    {
15        NewPipe();
16    }
17
18    // Update is called once per frame
19    void Update()
20    {
21        currentTime += Time.deltaTime;
22        if (currentTime > timeBetween)
23        {
24            NewPipe();
25            currentTime = 0f;
26        }
27    }
28
29    void NewPipe()
30    {
31        Vector3 newPosition = new Vector3(transform.position.x, transform.position.y + Random.Range(-0.6f, 1.6f), transform.position.z);
32        Instantiate(pipes, newPosition, transform.rotation);
33    }
34 }
```

Nouveau *Pipes* tous les 2 secondes

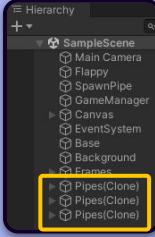
Instanciation d'un nouveau
GameObject Pipes

Offset aléatoire pour modifier la
hauteur du *Pipe*

Éviter une surcharge de la mémoire

en supprimant les *GameObjects Pipes* lorsqu'ils ne sont plus utiles

Ajouter une condition pour supprimer au fur et à mesure les *Pipes* générés qui disparaissent de l'écran

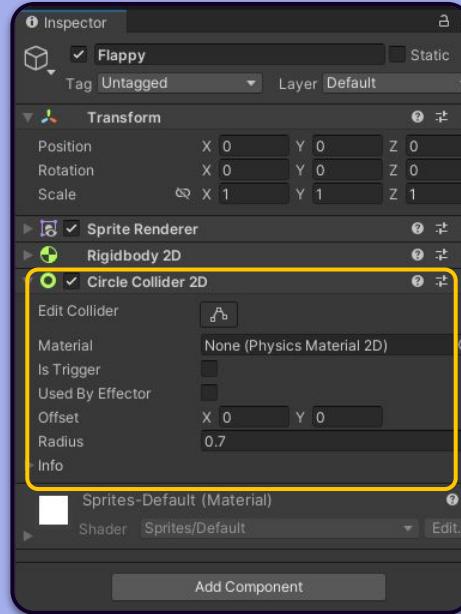


```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PipesMoveScript : MonoBehaviour
6  {
7      public float moveSpeed = 1f;
8
9      // Start is called before the first frame update
10     void Start()
11     {
12     }
13
14
15     // Update is called once per frame
16     void Update()
17     {
18         transform.position = transform.position + (Vector3.left * moveSpeed * Time.deltaTime);
19
20         if (transform.position.x < -2)
21         {
22             Destroy(gameObject);
23             Debug.Log("GameObject supprimé");
24         }
25     }
26 }
```

Affichage d'un message au sein de la console de Unity

Gérer les collisions

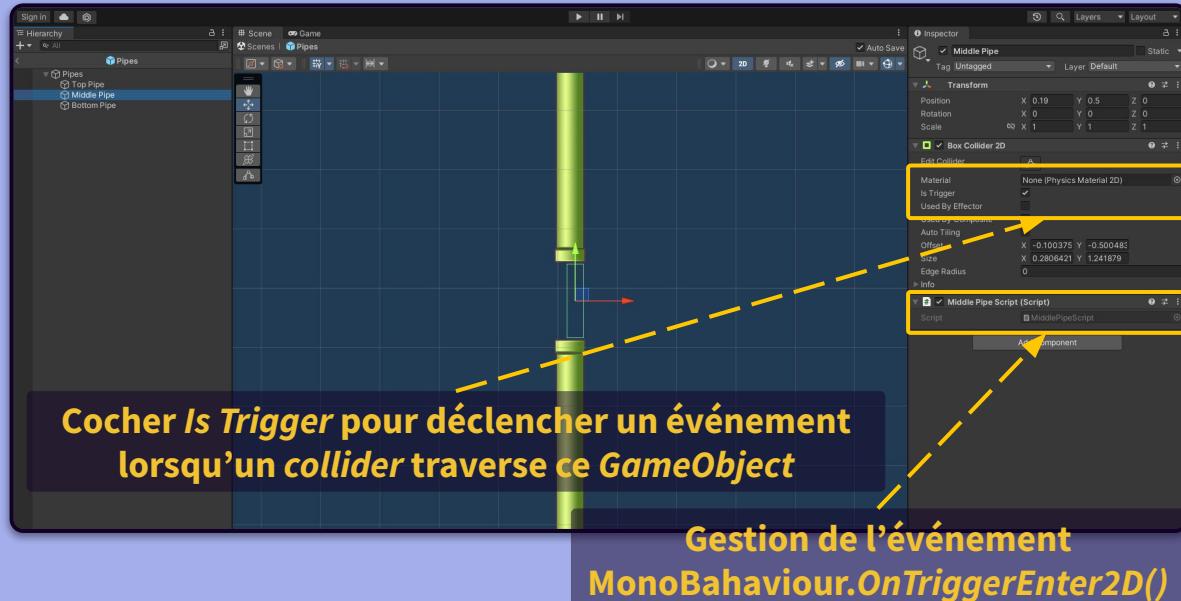
sélectionner *Flappy* et cliquer sur *Add component > Physics 2D > Circle Collider 2D*



Permet à votre *GameObject* d'interagir avec d'autres *colliders* (e.g. les tuyaux seront des *colliders*)

Créer une zone pour détecter le passage de Flappy en modifiant la *prefab* de Pipes

Créer un nouveau *GameObject* *Middle Pipe* et lui ajouter un composant *Box Collider 2D* ainsi qu'un script pour gérer l'événement de collision



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class MiddlePipeScript : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start() {}
9
10    // Update is called once per frame
11    void Update() {}
12
13    void OnTriggerEnter2D(Collider2D collider)
14    {
15    }
16
17 }
```

Centralisation des données de l'application

au sein d'un *GameObject GameManager*

Créer un nouveau *GameObject* nommé *GameManager* et lui ajouter un nouveau script

Ce *GameManager* sera en charge de conserver le score et d'actualiser l'UI lorsque l'utilisateur gagne un nouveau point ou lorsqu'il déclenche le *gameover* du jeu

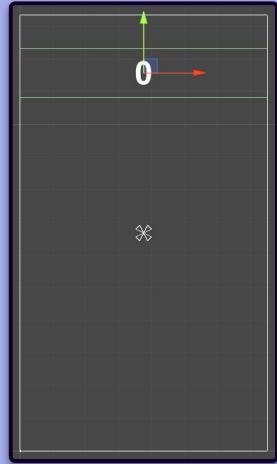
```
● ● ●
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Manager : MonoBehaviour
6  {
7      public float score = 0f;
8
9      // Start is called before the first frame update
10     void Start() {}
11
12     // Update is called once per frame
13     void Update() {}
14
15     public void AddPoint()
16     {
17         score += 1;
18     }
19
20     public void GameOver() {}
21 }
```

L'objectif est d'appeler cette fonction lorsque Flappy déclenche l'événement de Trigger

UI: Ajouter un score

clic droit au sein de la Hiérarchie > UI > Legacy > Text

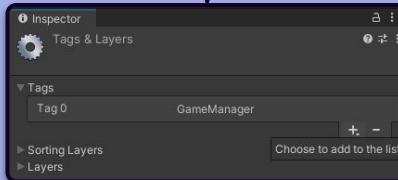
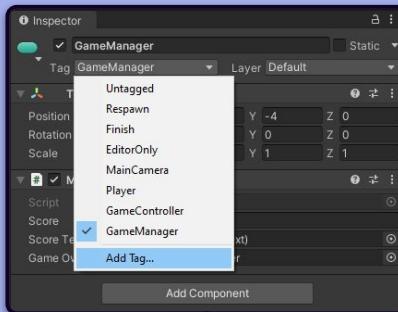
Lorsque vous sélectionnez un élément de la hiérarchie et que vous appuyez sur F, cela vous permet de centrer la vue sur l'élément en question (faites le pour l'élément Canvas)



Positionner le *Text* en haut et modifier ses propriétés

Utiliser un composant qui n'appartient pas au GameObject en utilisant le système de Tag

L'objectif est d'attribuer une étiquette à notre *GameManager* afin d'y faire référence au sein d'un autre composant qui ne lui appartient pas



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class MiddlePipeScript : MonoBehaviour
6 {
7     private Manager gameManager;
8
9     // Start is called before the first frame update
10    void Start()
11    {
12        gameManager = GameObject.FindGameObjectWithTag("GameManager").GetComponent<Manager>();
13    }
14
15    // Update is called once per frame
16    void Update()
17    {
18    }
19
20    void OnTriggerEnter2D(Collider2D collider)
21    {
22        gameManager.AddPoint();
23    }
24
25 }
```

The screenshot shows a C# script named 'MiddlePipeScript'. Two sections of the code are highlighted with yellow boxes. The first highlighted section is the 'Start' method, which retrieves the 'Manager' component from a GameObject with the tag 'GameManager'. The second highlighted section is the 'OnTriggerEnter2D' event handler, which calls the 'AddPoint' method on the 'gameManager' variable.

Utiliser un composant qui n'appartient pas au *GameObject*

en utilisant une alternative au système de *Tag*: le *design pattern Singleton*

Le *Singleton* est un *design pattern* permettant de manipuler des valeurs à travers l'ensemble du projet

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Manager : MonoBehaviour
6  {
7      public static Manager Instance;
8      public float score = 0f;
9
10     void Awake()
11     {
12         if (Instance == null)
13         {
14             Instance = this;
15         }
16     }
17     // Start is called before the first frame update
18     void Start() {}
19
20     void Update() {}
21
22     public void AddPoint()
23     {
24         score += 1;
25     }
26
27     public void GameOver() {}
28
29 }
```

Utilisation d'une propriété static

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class MiddlePipeScript : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start() {}
9
10     // Update is called once per frame
11     void Update() {}
12
13     void OnTriggerEnter2D(Collider2D collider)
14     {
15         Manager.Instance.AddPoint();
16     }
17 }
```

Actualisation de l'UI pour afficher le score

à l'aide d'une référence au *GameObject Text* au sein du *GameManager*

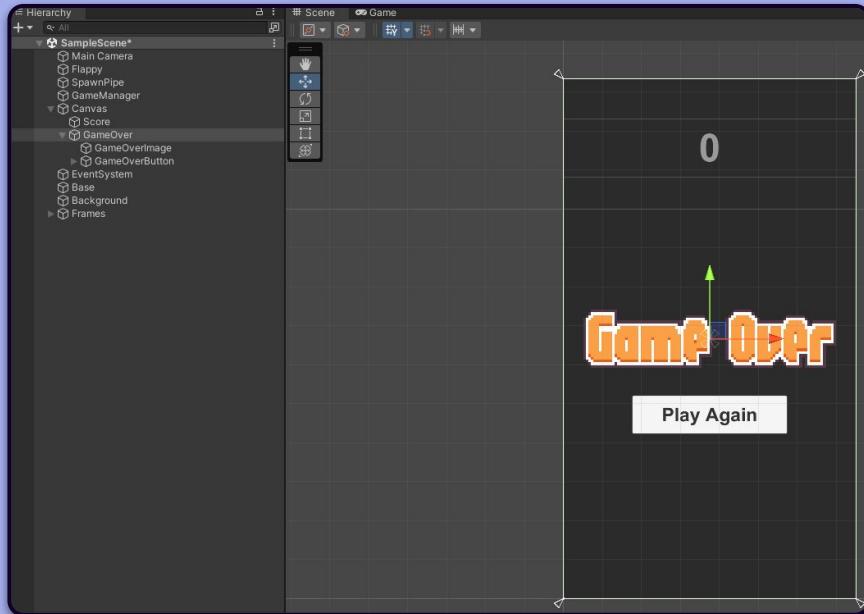
Créer une référence à votre *GameObject Text* au sein du script de votre *GameManager*

Importer la librairie *UnityEngine.UI* pour accéder à la classe *Text*

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class Manager : MonoBehaviour
7  {
8      public float score = 0f;
9      public Text scoreText;
10
11     // Start is called before the first frame update
12     void Start() {}
13
14     // Update is called once per frame
15     void Update() {}
16
17     public void AddPoint()
18     {
19         score += 1;
20         scoreText.text = score.ToString();
21     }
22
23     public void GameOver() {}
24 }
```

Ajouter une condition de Game Over et un UI adéquat

un écran de Game Over apparaît lorsque Flappy percute un tuyau (ou sort de l'écran)



Modifier la prefab de *Pipes* en ajoutant un composant *BoxCollider2D* aux *GameObject* *Top* et *Bottom Pipe*

Ajouter des *BoxCollider2D* au dessus et en dessous de Flappy

Ajouter à votre Canvas un panel, *clic droit* sur le *Canvas* > *UI* > *Panel*

Ajouter l'image du *Game Over* au sein du *Panel*, *clic droit* sur le *Panel* > *UI* > *Image*

Ajouter un bouton pour relancer une partie, *clic droit* sur *Panel* > *UI* > *Legacy* > *Button*

Associer une fonction à l'événement de clic

en liant une méthode *public* d'un *GameObject* au *Button*

Ajouter une fonction pour relancer le jeu au sein du *GameManager*

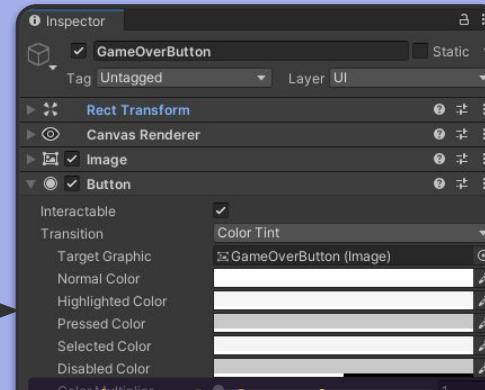
Importer la librairie *UnityEngine.SceneManagement* pour accéder à la classe *SceneManager*

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.SceneManagement;
6
7  public class Manager : MonoBehaviour
8  {
9      public float score = 0f;
10
11     public Text scoreText;
12     public GameObject gameOverPanel;
13
14     // Start is called before the first frame update
15     void Start() {}
16
17     // Update is called once per frame
18     void Update() {}
19
20     public void AddPoint()
21     {
22         score += 1;
23         scoreText.text = score.ToString();
24     }
25
26     public void ResetScene()
27     {
28         SceneManager.LoadScene(SceneManager.GetActiveScene().name);
29         gameOverPanel.SetActive(false);
30     }
31
32     public void GameOver()
33     {
34         gameOverPanel.SetActive(true);
35     }
36 }
```

Référence au Panel de Game Over

Rechargement de la scène courante

Activation du Panel associé au Game Over (désactivé par défaut)



Attribuer la fonction *ResetScene()* à l'événement de clic du bouton

Déclencher le Game Over lors d'une collision

à l'aide de la méthode *OnCollisionEnter2D()* de la classe *MonoBehaviour*

Implémenter la fonction *MonoBehaviour.OnCollisionEnter2D()* pour le *GameObject* de Flappy afin de déclencher un événement lorsqu'il percute un autre *collider*



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class FlappyScript : MonoBehaviour
6  {
7      public float flyMove = 5f;
8      public Rigidbody2D myRigidbody;
9      private Manager gameManager;
10     private bool birdAlive = true;
11
12     // Start is called before the first frame update
13     void Start()
14     {
15         gameManager = GameObject.FindGameObjectWithTag("GameManager").GetComponent<Manager>();
16     }
17
18     // Update is called once per frame
19     void Update()
20     {
21         if (Input.GetKeyDown(KeyCode.Space) && birdAlive)
22         {
23             myRigidbody.velocity = Vector2.up * flyMove;
24         }
25     }
26
27     void OnCollisionEnter2D(Collision2D collision)
28     {
29         birdAlive = false;
30         gameManager.GameOver();
31     }
32 }
```

Référence au *GameManager*

Attribut pour représenter l'état de Flappy

Enlever le contrôle du joueur lors du
GameOver

Déclencher la méthode *GameOver()* du
GameManager lors d'une collision

Animation et Son

en utilisant la fenêtre *Animation* et le composant *AudioSource*



Animation de Flappy

à l'aide du composant *Animator* et de la fenêtre *Animation*

Importer les images restantes de Flappy: *yellowbird-downflap.png* et *yellowbird-upflap.png*

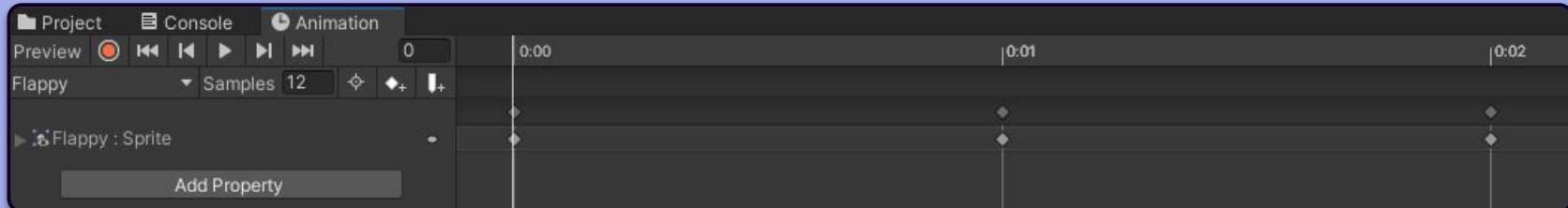
Clic sur Flappy et *Window > Animation > Animation* pour ouvrir la fenêtre liée à la création d'animation

Cliquer sur *Create* et indiquer un dossier pour sauvegarder votre animation

Un composant *Animator* s'ajoute automatiquement à Flappy

Sélectionner vos 3 images de Flappy et les glisser-déposer dans la fenêtre *Animation*

Modifier le *Sample* de l'animation à 12



Ajouter un son lors d'une collision

à l'aide du composant *AudioSource*

Importer le son *hit.wav*

Ajouter le composant *AudioSource* à Flappy, renseigner la propriété *AudioClip* par le son importé et désactiver *Play On Awake*

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class FlappyScript : MonoBehaviour
6  {
7      public float flyMove = 5f;
8      public Rigidbody2D myRigidbody;
9      private Manager gameManager;
10     private bool birdAlive = true;
11     private AudioSource _audioSource;
12
13     // Start is called before the first frame update
14     void Start()
15     {
16         gameManager = GameObject.FindGameObjectWithTag("GameManager").GetComponent<Manager>();
17         _audioSource = GetComponent<AudioSource>();
18     }
19
20     // Update is called once per frame
21     void Update()
22     {
23         if (Input.GetKeyDown(KeyCode.Space) && birdAlive)
24         {
25             myRigidbody.velocity = Vector2.up * flyMove;
26         }
27     }
28
29     void OnCollisionEnter2D(Collision2D collision)
30     {
31         if (birdAlive)
32         {
33             _audioSource.Play();
34         }
35         birdAlive = false;
36         gameManager.GameOver();
37     }
38 }
```

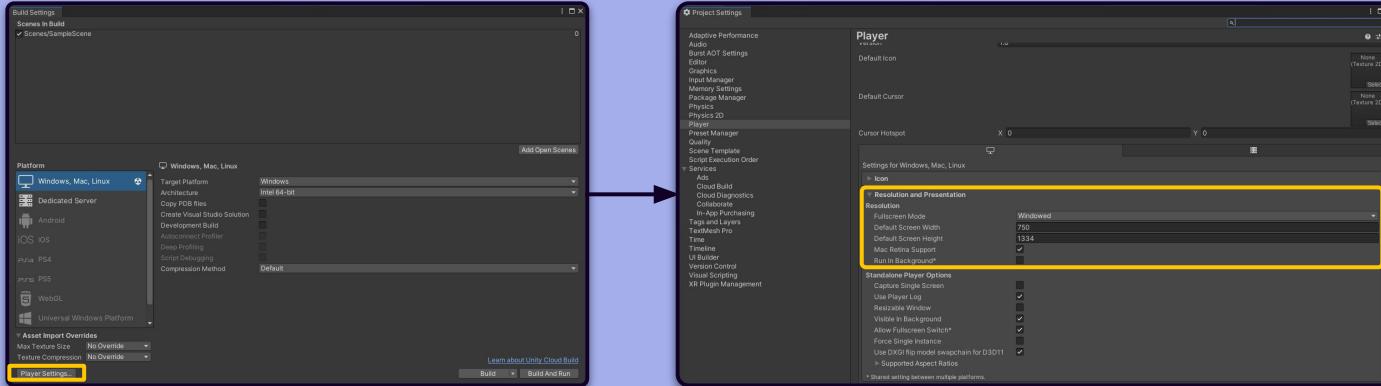
Référence au composant *AudioSource*

Exécute le son lors de la collision

Créer un exécutable de votre application

clic sur *File > Build Settings*

Modifier les *Player Settings* afin de modifier le mode d'affichage par *Windowed* et d'indiquer une taille par défaut (750x1334)



Cliquer ensuite sur le bouton *Build And Run* et choisissez un dossier dans lequel sauvegarder votre application

Selon la résolution de votre écran, la fenêtre de votre application peut ne pas avoir les bonnes dimensions. Vous pouvez alors ajouter la ligne suivante au sein de la méthode *Start()* de votre *GameManager* `Screen.SetResolution(500, 900, FullScreenMode.Windowed)`. La résolution indiquée doit être adaptée à celle de votre ordinateur.

Exercice pratique

à vous de jouer en implémentant une fonctionnalité supplémentaire



Implémenter un bonus en forme de cœur qui apparaît de manière aléatoire (e.g. 25% de chance) et qui augmente le score de +5 points si Flappy le percute

Ce bonus apparaît à une hauteur plus ou moins élevée en fonction de l'ouverture des tuyaux

Lorsque Flappy percute le bonus, jouer le son *point.wav* et ne plus afficher le bonus à l'écran